Instituto Superior Técnico

Applications and Computation for the Internet of Things
2019-2020

2nd Lab work: Sensing the Physical World

Group:		
Student 1:		
Student 2:		

Goal:

The goal of this work is to sense physical quantities and to control actuators according to the measured environment.

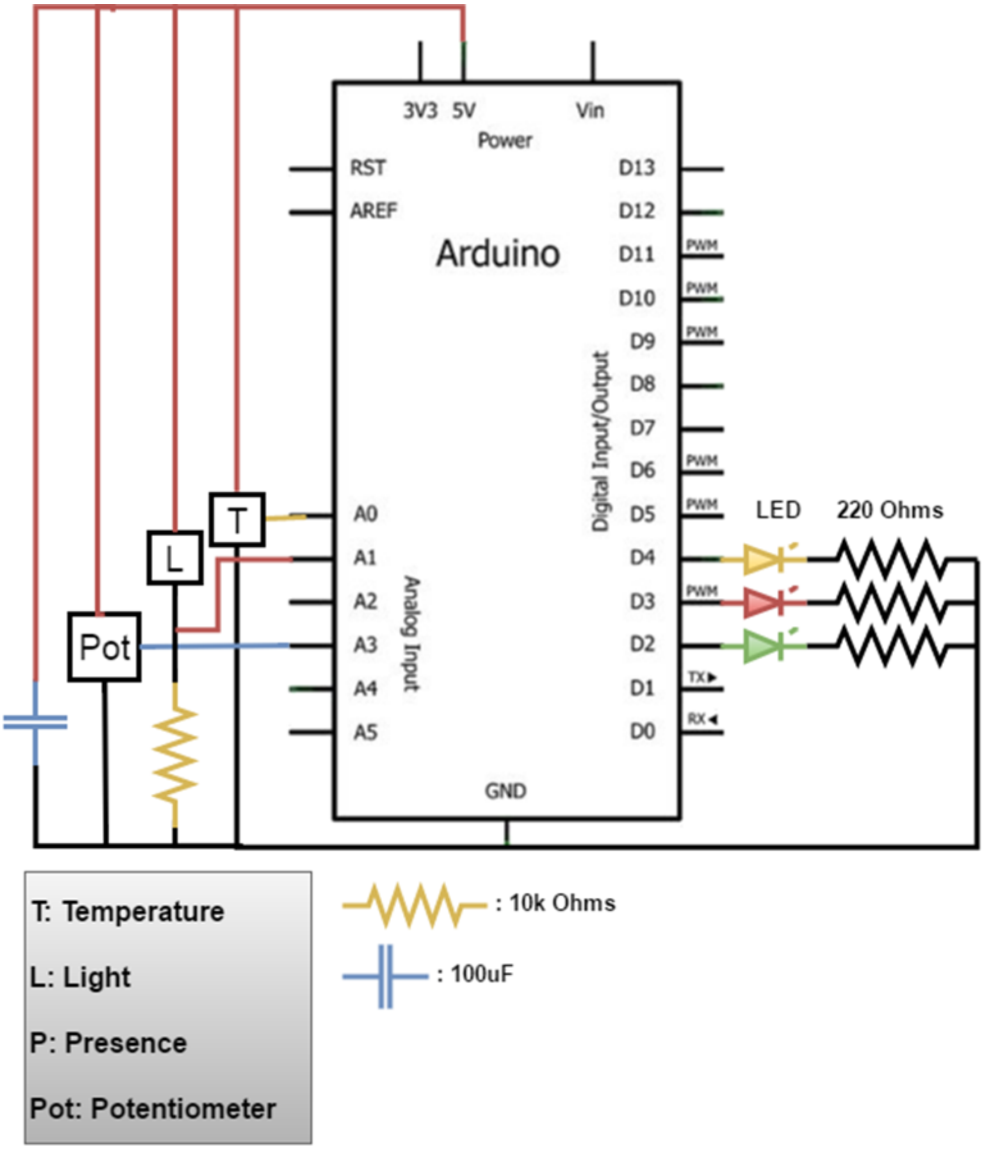
Description

Build an embedded system using an Arduino UNO board to simultaneously control 3 LEDs (the actuators) depending on the state of 3 different sensors – temperature, rotation angle (potentiometer) and light intensity.

- **Temperature** – The yellow LED associated with the temperature sensor must be turned on when the temperature read is greater than 26 ºC (value to be eventually redefined at the laboratory according to the environment conditions).
- **Rotation** – The green LED controlled by the potentiometer must blink with a period of time varying continuously between 0.2 and 2 seconds, depending of the rotation angle applied to the potentiometer.
- **Light** – The red LED for the light intensity function must change continuously its own light intensity based on the light intensity sensed in the surrounding environment from
  - Darkness → LED fully ON, to
  - Brightness → LED OFF

  (see Reference 5 about Pulse Width Modulation).

A diagram of the circuit is represented in the figure.



(in this work ignore “P: Presence”)

References

1. https://www.arduino.cc/en/Reference/digitalWrite
2. https://www.arduino.cc/en/Reference/AnalogRead
3. https://www.arduino.cc/en/Reference/Serial
4. https://www.arduino.cc/en/Tutorial/Calibration
5. https://www.arduino.cc/en/Tutorial/PWM
6. https://www.arduino.cc/en/Reference/Delay

Recommendation

In order to fulfill your work with security and not damaging the hardware involved, remember to carry out the recommendations below. As you are working fill the boxes to be certain that you fulfill all security measures.

Always work with the circuit disconnect from the source.	
Call the professor or responsible for the laboratory, before you connect the circuit to the source.	
Make sure the circuit is well connected (resistors, capacitors, etc.) to prevent a short circuit, or damage the hardware.	

Mapping analog measurements

Usually the digital readings retrieved from sensors do not correspond directly to the value of the physical quantity, but rather to values between 0 and a maximum binary value (such as, for a 10 bits word, $1023 = 2^{10} - 1$). Therefore some mapping may be required.

When the value is relative just to an offset a simple mapping is adequate, but in general a more complex scale conversion will be needed. The `map` function, available in the Arduino IDE, simplifies these types of conversions. For example, when rotating a Servo motor with input from a potentiometer we know that when the value of the potentiometer is 0 then the servo angle must be 0° as well. Logically, if the value of the potentiometer is 1023 (its maximum reading) then the servo angle must be 180° (its maximum position):

```
map(value, 0, 1023, 0, 180)
```

Besides this some sensors require a linearization of its transfer function (physical quantity → electrical quantity, such as voltage). For example, to convert the reading from a circuit with a temperature sensor (in our case a temperature dependent resistor) to the real temperature several transformations may be required

- to linearize the transfer function of the sensing device $R_T = f(T)$, and
- to linearize the transfer function of the circuit in which the sensor is included.

For the device and configuration to be used (based on a TMP35 component) check the function

$$T = (((sensor\ value \,/\, 1024.0) * 5.0\,) - 0.5\,) * 100$$

Programming with analog sensors:

To access an external analog sensor you must attach it to an Arduino analog pin. The software allocation of a sensor to an analog pin is done using the following code:

```
    int const tempSensor = A0;
```

where `A0` is the physical pin where the sensor is attached.

To read the value assigned to a specific pin use the Arduino function `analogRead(PIN)`, as follows:

$$int\ temperatureValue\ =\ analogRead(tempSensor);$$

Debug:

In order to control some variables and *debug* your program it is possible to print them to the Serial Monitor available in the Arduino IDE. This kind of tool is useful, for example, to keep track of the temperature variation or to help the calibration of the system.

To use this feature, first start a serial communication with the PC:

$$void\ setup()\ \{\ \ldots;\ \ Serial.begin(9600);\ \ \ldots;\}$$

Then, just *Serial.print* your variables and/or strings:

$$Serial.println(temperatureValue);$$

Program the application:

Provide your program listing.

Structure the program modularly clearly segmenting the program in separate code blocks, one for each function to be implemented (e. g. read temperature sensor, control temperature alarm). In the next laboratory work you will have to distribute several functions across separate Arduino controller.

Avoid, or use very carefully the delay function provided in the Arduino IDE since it just stops the execution flow during the specified time interval.

- Code:

```
void setup() {


}

void loop() {



}
```

- Questions:

Question:

For each of the three pairs sensor-actuator describe:

1. the mapping process implemented;
2. the calibration setup;
3. the process, or technique, used to modulate the behavior of the actuator;
4. the setup prepared to demonstrate the functionality of the system.

Answer:

Temperate:

Rotation:

Light:

Questions:

What is the system software pattern of the application?

What are the timing constraints of the system?